Knowledge bases

- Knowledge base = set of **sentences** in a **formal** language

- **Declarative** approach to building an agent (or other system):
  - **Tell** it what it needs to know
  - Then it can **Ask** itself what to do - answers should follow from the KB
  - **Tell** it what action the agent will take

- Agents can be viewed at the **knowledge level**
  i.e., what they know, regardless of how implemented

- Or at the **implementation level**
  i.e., data structures in KB and algorithms that manipulate them
A simple knowledge-based agent

```
function KB-AGENT( percept) returns an action
    static: KB, a knowledge base
              t, a counter, initially 0, indicating time
    TELL(KB, MAKE-PERCEPT-SENTENCE( percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

- The agent must be able to:
  - Represent states, actions, etc.
  - Incorporate new percepts
  - Update internal representations of the world
  - Deduce hidden properties of the world
  - Deduce appropriate actions
Wumpus World PEAS description

- **Performance measure**
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow

- **Environment**
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square

- **Sensors:** Stench, Breeze, Glitter, Bump, Scream

- **Actuators:** Left turn, Right turn, Forward, Grab, Release, Shoot
Wumpus world characterization

- **Fully Observable** No – only local perception
- **Deterministic** Yes – outcomes exactly specified
- **Episodic** No – sequential at the level of actions
- **Static** Yes – Wumpus and Pits do not move
- **Discrete** Yes
- **Single-agent?** Yes – Wumpus is essentially a natural feature
Exploring a wumpus world
Exploring a wumpus world
Exploring a wumpus world
Exploring a wumpus world
Exploring a wumpus world
Exploring a wumpus world
Exploring a wumpus world
Exploring a wumpus world
What is logic?

• **Logic** is a formal system for manipulating facts so that true conclusions may be drawn.

• **Syntax:** rules for constructing valid sentences
  - E.g., \( x + 2 \geq y \) is a valid arithmetic sentence, \( x2y + \text{ is not} \)

• **Semantics:** “meaning” of sentences, or relationship between logical sentences and the real world
  - Specifically, semantics defines truth of sentences
  - E.g., \( x + 2 \geq y \) is true in a world where \( x = 5 \) and \( y = 7 \)
Propositional logic: Syntax

• Propositional logic is the simplest logic – illustrates basic ideas

• The proposition symbols $P_1, P_2, \text{True}, \text{False}, \text{etc}$ are sentences

  • If $S$ is a sentence, $\neg S$ is a sentence (negation)

  • If $S_1$ and $S_2$ are sentences, $S_1 \land S_2$ is a sentence (conjunction)

  • If $S_1$ and $S_2$ are sentences, $S_1 \lor S_2$ is a sentence (disjunction)

  • If $S_1$ and $S_2$ are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)

  • If $S_1$ and $S_2$ are sentences, $S_1 \iff S_2$ is a sentence (biconditional)
Propositional logic: Semantics

• A **model** specifies the true/false status of each proposition symbol in the knowledge base
  • E.g., \( P \) is **true**, \( Q \) is **true**, \( R \) is **false**
  • With three symbols, there are 8 possible models, and they can be enumerated exhaustively

• Rules for evaluating truth with respect to a model:

\[
\neg P \quad \text{is true} \quad \text{iff} \quad P \quad \text{is false}
\]
\[
P \land Q \quad \text{is true} \quad \text{iff} \quad P \quad \text{is true} \quad \text{and} \quad Q \quad \text{is true}
\]
\[
P \lor Q \quad \text{is true} \quad \text{iff} \quad P \quad \text{is true} \quad \text{or} \quad Q \quad \text{is true}
\]
\[
P \Rightarrow Q \quad \text{is true} \quad \text{iff} \quad P \quad \text{is false} \quad \text{or} \quad Q \quad \text{is true}
\]
\[
P \iff Q \quad \text{is true} \quad \text{iff} \quad P \Rightarrow Q \quad \text{is true} \quad \text{and} \quad Q \Rightarrow P \quad \text{is true}
\]
Truth tables

- A **truth table** specifies the truth value of a composite sentence for each possible assignments of truth values to its atoms.

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( P )</td>
<td>( Q )</td>
<td>( \neg P )</td>
<td>( P \land Q )</td>
<td>( P \lor Q )</td>
<td>( P \Rightarrow Q )</td>
<td>( P \iff Q )</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

- The truth value of a more complex sentence can be evaluated *recursively* or *compositionally*.
**Logical equivalence**

- Two sentences are **logically equivalent** iff true in same models: \( \alpha \equiv \beta \) iff \( \alpha \models \beta \) and \( \beta \models \alpha \)

\[
\begin{align*}
(\alpha \land \beta) & \equiv (\beta \land \alpha) \quad \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor \\
\neg(\neg \alpha) & \equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv (\neg \beta \Rightarrow \neg \alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv (\neg \alpha \lor \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv (\neg \alpha \lor \neg \beta) \quad \text{de Morgan} \\
\neg(\alpha \lor \beta) & \equiv (\neg \alpha \land \neg \beta) \quad \text{de Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land
\end{align*}
\]
Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

$\neg P_{1,1}$
$\neg B_{1,1}$
$B_{2,1}$

- "Pits cause breezes in adjacent squares"

$B_{1,1} \iff (P_{1,2} \lor P_{2,1})$
$B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})$
Truth tables for inference

<table>
<thead>
<tr>
<th>$B_{1,1}$</th>
<th>$B_{2,1}$</th>
<th>$P_{1,1}$</th>
<th>$P_{1,2}$</th>
<th>$P_{2,1}$</th>
<th>$P_{2,2}$</th>
<th>$P_{3,1}$</th>
<th>$KB$</th>
<th>$\alpha_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>
Validity and satisfiability

A sentence is **valid** if it is true in **all** models,
  e.g., $\text{True}$, $A \lor \neg A$, $A \Rightarrow A$, $(A \land (A \Rightarrow B)) \Rightarrow B$

A sentence is **satisfiable** if it is true in **some** model
  e.g., $A \lor B$, $C$

A sentence is **unsatisfiable** if it is true in **no** models
  e.g., $A \land \neg A$

Satisfiability is connected to inference via the following:

$KB \models \alpha$ if and only if $(KB \land \neg \alpha)$ is unsatisfiable
Entailment

• **Entailment** means that a sentence follows from the premises contained in the knowledge base:

\[ KB \models \alpha \]

• Knowledge base \( KB \) entails sentence \( \alpha \) if and only if \( \alpha \) is true in all models where \( KB \) is true
  • E.g., \( x = 0 \) entails \( x \times y = 0 \)

• \( KB \models \alpha \) iff \( (KB \Rightarrow \alpha) \) is valid
• \( KB \models \alpha \) iff \( (KB \land \neg \alpha) \) is unsatisfiable
Inference

- $KB \models_i \alpha = \text{sentence } \alpha \text{ can be derived from } KB \text{ by procedure } i$

- **Soundness**: $i$ is sound if whenever $KB \models_i \alpha$, it is also true that $KB \models \alpha$

- **Completeness**: $i$ is complete if whenever $KB \models \alpha$, it is also true that $KB \models_i \alpha$

- Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

- That is, the procedure will answer any question whose answer follows from what is known by the $KB$. 
Inference

• How can we check whether a sentence $\alpha$ is entailed by KB?
• How about we enumerate all possible models of the KB (truth assignments of all its symbols), and check that $\alpha$ is true in every model in which KB is true?
  • Is this sound?
  • Is this complete?
• Problem: if KB contains $n$ symbols, the truth table will be of size $2^n$
• Better idea: use inference rules, or sound procedures to generate new sentences or conclusions given the premises in the KB
Proof methods

- Proof methods divide into (roughly) two kinds:
  - Application of inference rules
    - Legitimate (sound) generation of new sentences from old
    - **Proof** = a sequence of inference rule applications
      Can use inference rules as operators in a standard search algorithm
    - Typically require transformation of sentences into a normal form
  - Model checking
    - truth table enumeration (always exponential in \( n \))
    - heuristic search in model space (sound but incomplete)
      e.g., min-conflicts-like hill-climbing algorithms
Inference rules

- Modus Ponens

\[ \alpha \Rightarrow \beta, \alpha \]
\[ \beta \]

- And-elimination

\[ \alpha \land \beta \]
\[ \alpha \]
Inference rules

- And-introduction

\[
\frac{\alpha, \beta}{\alpha \land \beta}
\]

- Or-introduction

\[
\frac{\alpha}{\alpha \lor \beta}
\]
Inference rules

- Double negative elimination

\[
\begin{align*}
\neg\neg\neg\neg\alpha \\
\hline
\alpha \\
\hline
\alpha
\end{align*}
\]

- Unit resolution

\[
\begin{align*}
\alpha \lor \beta, \neg\beta \\
\hline
\alpha
\end{align*}
\]
Resolution

\[
\frac{\alpha \lor \beta, \neg \beta \lor \gamma}{\alpha \lor \gamma}
\]

or

\[
\frac{\alpha \lor \beta, \beta \Rightarrow \gamma}{\alpha \lor \gamma}
\]

• Example:
  \(\alpha\): “The weather is dry”
  \(\beta\): “The weather is rainy”
  \(\gamma\): “I carry an umbrella”
Resolution is complete

\[
\alpha \lor \beta, \neg \beta \lor \gamma \\
\hline
\alpha \lor \gamma
\]

- To prove \( KB \models \alpha \), assume \( KB \land \neg \alpha \) and derive a contradiction
- Rewrite \( KB \land \neg \alpha \) as a conjunction of clauses, or disjunctions of literals
  - Conjunctive normal form (CNF)
- Keep applying resolution to clauses that contain complementary literals and adding resulting clauses to the list
  - If there are no new clauses to be added, then \( KB \) does not entail \( \alpha \)
  - If two clauses resolve to form an empty clause, we have a contradiction and \( KB \models \alpha \)
Complexity of inference

• Propositional inference is _co-NP-complete_

• *Complement* of the SAT problem: $\alpha \vdash \beta$ if and only if the sentence $\alpha \land \neg \beta$ is *unsatisfiable*

• Every known inference algorithm has worst-case exponential running time

• Efficient inference possible for restricted cases